Mitigating Both Data Corruption and Content Replay Attacks with Implicit Data Integrity

Michael Kounavis
Intel Labs, Intel Corporation, 2111, NE 25th Avenue, Hillsboro, OR 97124

Email: michael.e.kounavis@intel.com

***Abstract*—We study the security of the recently proposed implicit integrity methodology. Implicit integrity is a novel methodology that supports corruption detection without producing, storing or verifying mathematical summaries of the content such as MACs or ICVs, as typically done today. The main idea behind implicit integrity is that, whereas typical user data demonstrate patterns such as repeated bytes or words, decrypted data resulting from corrupted ciphertexts no longer demonstrate such patterns. Thus, by checking the entropy of decrypted ciphertexts, corruption can be possibly detected.**

Past contributions to the implicit integrity methodology have focused on observed patterns on client and server data that motivate the methodology, entropy definitions for arbitrarily small messages, and constructions that mitigate data corruption attacks. In this paper, we extend the known analytical results concerning implicit integrity addressing content replay attacks as well. We demonstrate that the class of cryptographic constructions known as 'random oracles according to observer functions', which has been proposed for mitigating data corruption attacks, is actually simultaneously secure under two different adversary models: an input perturbing adversary performing content corruption attacks, and an oracle replacing adversary performing content replay attacks.

I. Introduction

We address the problem of detecting both data corruption and content replay without producing, storing or verifying mathematical summaries of the content. Such summaries known as Message Authentication Codes (MACs) [6] [7] or Integrity Check Values (ICVs) are typically costly to maintain and use. The standard way of supporting data integrity is by using MACs produced by cryptographic hash functions such as SHA256 [4] or SHA3 [5], the use of which in many cases results in latency, storage and communication overheads. These overheads are due to the unavoidable message expansion associated with using the MACs.

The paper studies the security of an alternative methodology that uses pattern techniques in order to support corruption detection for the large majority of user data without message expansion. The main idea is shown in figure 1. If some content exhibits patterns (i.e., has low entropy), then such content can be distinguished from random data. Let's consider that this content is encrypted, as shown in the figure, where the encryption algorithm is a good pseudo-random permutation and thus can successfully approximate a random oracle. The ciphertext which is produced in this way is no longer distinguishable from random data, under certain reasonable assumptions about the adversary. Any corruption

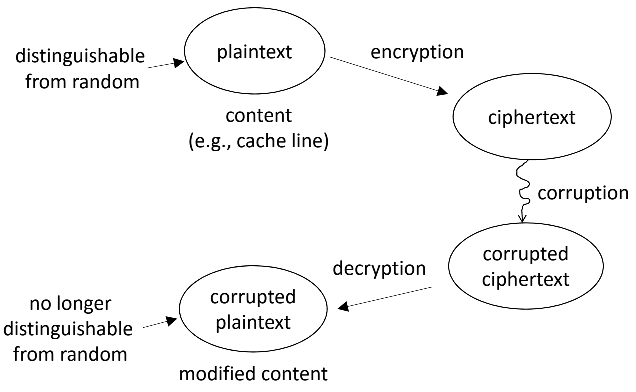

Fig. 1. The concept of implicit integrity

on the ciphertext results in a new ciphertext value, which is different from the original one. Furthermore, any decryption operation on this new ciphertext value results in a corrupted plaintext value which is different from the original one as well. As decryption is the inverse operation of encryption, the decryption algorithm also approximates a random oracle under reasonable adversary models. Because of this reason, the corrupted plaintext value is also indistinguishable from random data with very high probability. For these reasons, checking the entropy of the result of a decryption operation can be a reliable test for detecting corruption for some data. References [1] [2] [3] refer to such methodology as 'implicit data integrity' or just 'implicit integrity'.

The fact that uncompressed, unencrypted user data demonstrate patterns should not come as surprise. User data often consist of media data, tables, code, data structures, and other types of structured data that are characterized by significant redundancy. For example, there exists a simple pattern which is frequently encountered in client and server data. This is the appearance of 4 or more 16-bit words which are equal to each other in a collection of 32 words. According to the experimental observations reported in [2] and [3], coming from over 111 million client cache lines and 1.47 billion server cache lines, such pattern characterizes 82% of the client data and 78% of the server data.

To study implicit integrity, reference [3] introduces a class of constructions referred to as 'Random Oracles according to

Observer functions' (RO^2). An observer function is a function that searches the output of cryptographic systems in order to detect unusual behavior, such as the presence of patterns. Patterns can be repeated nibbles, bytes, words or double words. If an observer function detects unusual behavior with the same or similar probability in a cryptographic system's output as in a random oracle's output then such cryptographic system belongs to the class of RO^2 constructions associated with the specific observer function.

The main contribution of this paper is the description of a security model against both data corruption and replay attacks, which is associated with implicit integrity and is connected with the class of RO^2 constructions. Specifically, we show that if a construction is in the RO^2 class, then the construction always supports some form of implicit data integrity, and is secure in the proposed model. The proposed model comprises two kinds of adversaries.

First, an input perturbing adversary is an algorithm which is given a set of ciphertext messages $q_0, \ldots, q_{m-1}$, the plaintexts of which exhibit patterns and a query bound B. The algorithm succeeds if it finds a ciphertext message y which is different from $q_0, \ldots, q_{m-1}$, the plaintext of which also exhibits patterns. It is considered that the encryption key is unknown. Security in this adversary model indicates protection against data corruption attacks.

Second, an oracle replacing adversary is an algorithm which is also given a set of ciphertext messages $q_0, \ldots, q_{m-1}$ the plaintexts of which exhibit patterns and a query bound B. The algorithm succeeds if it replaces a set of internal oracles $R_0, R_1, \ldots$ queried by the decryption system with a new set of internal oracles $R'_0, R'_1, \ldots$, so that there exists a ciphertext message $y \in \{q_0, \ldots, q_{m-1}\}$ the plaintext of which continues to exhibit patterns even when the oracles $R_0, R_1, \ldots$ of the decryption system are replaced by $R'_0, R'_1, \ldots$. We discuss that security in this adversary model indicates protection against content replay attacks. Content replay attacks are considered across key domains, where the associated encryption keys of these key domains are unknown.

II. Related Work

Implicit integrity is introduced in reference [1]. Past contributions to the implicit integrity methodology have either focused on the entropy properties of client and server data [2] and entropy definitions suitable for arbitrarily small messages, or on constructions that mitigate data corruption attacks [3]. Our paper extends these result substantially discussing replay attacks as well.

Other contributions to the similar concept or robust authenticated encryption [8], [9] base their notion of security on the indifferentiability or indistinguishability of their proposed constructions from a random permutation or a random function. Indifferentiability and indistinguishability are mostly used in these references in the general sense (as in reference [10]). For example reference [8] proposes a wide block cipher design (AEZ) and proves that this design is indeed difficult to distinguish from a wide random permutation. While such notion of security is useful, and the analysis of references [8], [9] insightful, implicit integrity is not associated with such strong notion of security.

One aspect of the methodology introduced in reference [3], which we also follow in this paper, is that the derivation of any security statements concerning integrity is decoupled from the derivations of statements concerning confidentiality. The constructions studied in this paper support confidentiality in the well established pseudo-random permutation (PRP) model, as their output is produced by a set of of pseudo-random permutations. The constructions also support implicit integrity according to the notion of security introduced in [3]. This notion of security reflects the fact that it should be computationally difficult for an adversary to corrupt some ciphertext, so that the resulting plaintext demonstrates specific patterns.

III. Preliminary Concepts

We begin our discussion with the concept random oracles according to observer functions introduced in [3]. We consider 'observer functions' that search cryptographic system outputs in order to detect 'abnormal behavior', such as repetitions of values of different sizes.

Random oracles according to observer functions are constructions which may be distinguishable from ideal primitives, but appear indistinguishable only with respect to *specific* distinguishers. The concept is illustrated in Figure 2. Function f observes unusual behavior in the values of the output of a random oracle with probability P_f. The same function f observes the same unusual behavior in the values of the output of the real system S with probability $P_{f,2}$. If $P_{f,2} \leq P_f \cdot 2^{\epsilon}$ for a given maximum *non-repeating* input sequence of size B, and if this relation holds even when $P_{f,2}$ is conditioned upon previous inputs, then the system S is a "random oracle according to observer function f" associated with an indistinguishability parameter ϵ: $S \in \mathbf{RO}^2(f, B, \epsilon)$.

The query bound B denotes the life time of the construction. A finite life time B is introduced in the definition of a RO^2 construction, so that the construction can contain primitives which are bijective functions (pseudo-random permutations) and which within the bounds of the life time B are indistinguishable from truncated output random oracles. These primitives are the pseudo-random permutations that encrypt and decrypt data. Furthermore it is these primitives that support data confidentiality. We will be denoting these as 'ingredient random permutations'.

In order for a construction to be RO^2 it needs to satisfy the condition $P_{f,2} \leq P_f \cdot 2^{\epsilon}$ for non-repeating input. So what does non-repeating input mean? Having non-repeating input means that:

- within a lifetime of a construction $\{y_0, y_1, \ldots, y_{B-1}\}$ input is not repeating, i.e., $y_i \neq y_j \; \forall i, j \in [0, B-1]$; and
- inputs that result in unusual behavior in one lifetime $\{y_0, y_1, \ldots, y_{B-1}\}$ of a construction are not repeated

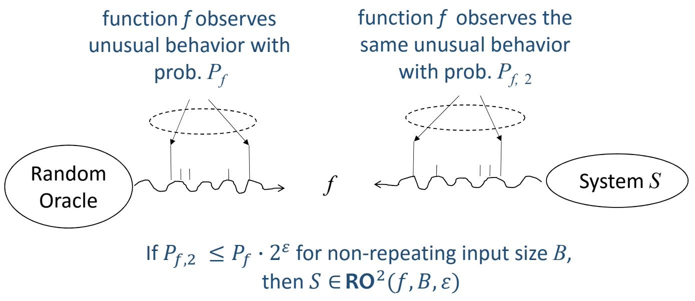
function f observes unusual behavior with prob. P_f
function f observes the same unusual behavior with prob. $P_{f,2}$
Random Oracle
f
System S
If $P_{f,2} \leq P_f \cdot 2^{\varepsilon}$ for non-repeating input size B, then $S \in \mathbf{RO}^2(f, B, \varepsilon)$

Fig. 2. The Concept of a Random Oracle according to an Observer function (RO^2)

in any other lifetime $\{z_0, z_1, \ldots, z_{B-1}\}$ of the same construction.

RO^2 constructions are most useful when the inputs considered are adversary queries, i.e., corrupted ciphertext values. In this case, the conditions of non-repeating input are more meaningful. The intuition behind introducing the non-repeating input conditions, which are used in the derivation of our main results below, is that if an adversary repeats queries as part of the attack strategy, then the system provides the same output for these repeated queries. Specifically, we consider that there are no parameters, potentially randomizing the system which are out of the adversary's control. Under this assumption it is clear that it is not beneficial for an adversary to repeat queries which are unsuccessful, as their result will be the same. On the other hand, if some queries are successful, then the adversary does not need to repeat these queries, as the adversary possesses the knowledge about the impact of such queries. Because of these reasons, it is not restrictive to introduce the non-repeating input requirement, as we consider adversaries that do not repeat their queries to the construction. We also note that we do not restrict every input to the construction. We just restrict only the inputs for which the condition $P_{f,2} \leq P_f \cdot 2^{\epsilon}$ needs to be satisfied.

Definition 1: *Observer function.* A function f associated with input strings of length L, $f : \{0,1\}^L \to \{0,1\}$ is called an observer function if it outputs only one of two values 0 or 1. If for some input x, $f(x) = 1$, then we will be saying that input x demonstrates unusual behavior according to observer function f. We will also be denoting this fact as $x \in \mathbf{\Pi}(f)$.

Definition 2: *Random Oracle according to an Observer function* (RO^2). Let $\{y_0^{(0)}, y_1^{(0)}, \ldots, y_{m_0-1}^{(0)}\}$, $\{y_0^{(1)}, y_1^{(1)}, \ldots, y_{m_1-1}^{(1)}\}, \ldots$ be sets of binary strings of length L, the cardinalities of which satisfy $m_i \leq B$, $\forall i \geq 0$. Let also f be an observer function associated with inputs of length L, and $R \leftarrow 2^{\infty}$ a random oracle. A function or system $S : \{0,1\}^L \to \{0,1\}^L$ is called a random oracle according to observer function f associated with a life time B and indistinguishability parameter ϵ if the following conditions are true:

i. $y_i^{(k)} \neq y_j^{(k)}$ for all $y_i^{(k)}, y_j^{(k)} \in \{y_0^{(k)}, y_1^{(k)}, \ldots, y_{m_k-1}^{(k)}\}$ such that $i \neq j$, $k \geq 0$;

ii. if $y \in \mathbf{\Pi}(f)$ and $y \in \{y_0^{(k)}, y_1^{(k)}, \ldots, y_{m_k-1}^{(k)}\}$ for some $k \geq 0$, then $y \notin \{y_0^{(l)}, y_1^{(l)}, \ldots, y_{m_l-1}^{(l)}\}$ for all $l \neq k$;

iii. for all inputs y and non-empty collections of inputs $y_0, \ldots, y_{q-1}$ such that $y \neq y_0, \ldots, y \neq y_{q-1}$, and $y \in \{y_0^{(k)}, y_1^{(k)}, \ldots, y_{m_k-1}^{(k)}\}$, $y_i \in \{y_0^{(l_i)}, y_1^{(l_i)}, \ldots, y_{m_{l_i}-1}^{(l_i)}\}$, $k \geq 0$, $l_i \geq 0$, $0 \leq i < q$, $q \geq 0$, the following is true:
Prob$[S(y) \in \mathbf{\Pi}(f) \mid y_0, \ldots, y_{q-1}] \leq P_f \cdot 2^{\epsilon}$ where $P_f =$ Prob$[trunc_L(R(y)) \in \mathbf{\Pi}(f)]$

where the function $trunc_L()$ used in Definition 2 truncates its input returning the input's L most significant bits.

When a system S is a random oracle according to an observer function f, life time B and indistinguishability parameter ϵ, we will be denoting this fact as $S \in \mathbf{RO}^2(f, B, \epsilon)$. We also note that condition (iii) in Definition 2 covers all cases where the probability of seeing unusual behavior in the output of S is conditioned upon any set of input values different from y of cardinality q. In the special case were $q = 0$ (i.e., no conditioning) this third condition is simplified as Prob$[S(y) \in \mathbf{\Pi}(f)] \leq P_f \cdot 2^{\epsilon}$. The probability value P_f will be denoted as 'observation probability' or 'pattern observation probability' associated with observer function f in this document.

Definition 3: *Constrained* RO^2 *function.* A 'constrained' RO^2 function or system uses a number of internal invertible functions which are random permutations and which, for the life time (i.e., query) bound B used, are practically indistinguishable from random oracles in both processing directions. These are the primitives referred to as ingredient random permutations. A constrained RO^2 system is invertible itself. One direction is denoted as E and referred to as 'encryption', whereas the other direction is denoted a D and referred to as 'decryption'. A constrained RO^2 system accepts a construction input y and uses a set of pre-processing, invertible, polynomial time algorithms $a_1, a_2, \ldots, a_n$ to compute the inputs to ingredient random permutations which provide a response vector r. Then, it is this response which is further used in the RO^2

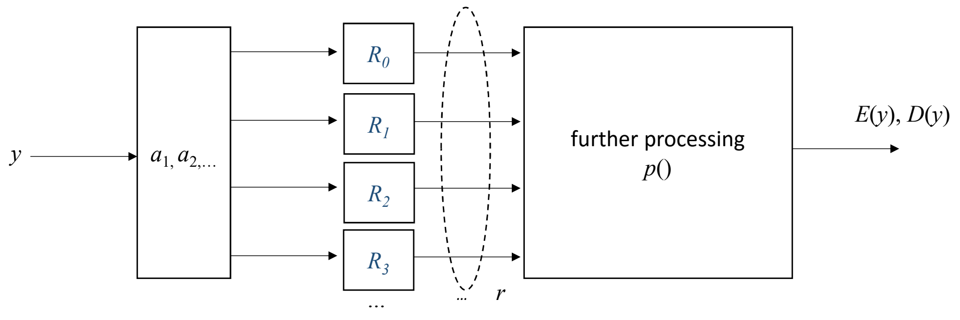

Fig. 3. Internal structure of a constrained RO^2 construction

system for processing. Processing is done by an invertible polynomial time algorithm $p()$ and the input y is no longer used. A constrained RO^2 system is illustrated in Figure 3. In what follows, whenever we will be using the term RO^2 we will be referring only to constrained RO^2 systems. We will also be using the notation $E^{R_0,R_1,R_2,\cdots}$ and $D^{R_0,R_1,R_2,\cdots}$ to refer to encryption and decryption systems (encryption and decryption oracles) with access to the ingredient random permutations $R_0, R_1, R_2, \ldots$. Furthermore if $E^{R_0,R_1,R_2,\cdots}$ and $D^{R_0,R_1,R_2,\cdots}$ are RO^2 associated with an observer function f, a query bound B and an indistinguishability parameter ϵ, we will be denoting this fact as:

$$E^{R_0,R_1,R_2,\cdots}, D^{R_0,R_1,R_2,\cdots} \in \mathbf{RO}^2(f, B, \epsilon) \qquad (1)$$

Proposition 1: The set of constrained RO^2 constructions is non-empty. Indeed, at least one construction has been proposed and proven to be in this class. This is the IVP introduced in reference [3].

We further note that if algorithms $a_1, a_2, \ldots, a_n$ are replaced by the identity function and the data path of Figure 3 implements a decryption operation, then the ciphertext of such system is obtained directly from the output of random permutations $R_0, R_1, R_2, \ldots$. This means that the system does not compromise the confidentiality offered by $R_0, R_1, R_2, \ldots$ in any way, provided that the implementation of $p()$, does not use any secret information also used by the implementations of $R_0, R_1, R_2, \ldots$.

IV. Adversary models and main security claims

A. Input perturbing adversary

The first type of adversary presented, describes adversaries which aim in corrupting encrypted data that are stored somewhere, or are in transit, in such a way so that the corruptions pass undetected. Reference [3] to such type of adversary as 'input perturbing' adversary.

The input perturbing adversary models any software process or physical intruder that aims in intentional corruptions of data. The underlying assumption of this adversary model is that the adversary can access data only in their encrypted form. This adversary can corrupt ciphertext data in any possible way hoping that the corruptions will result in plaintexts with patterns, and thus pass undetected. This adversary model is not unrealistic. In secure network connections or encrypted storage systems, many attacks originate from sources outside of these trusted domains. These attackers can possibly inspect a range of encrypted data, such as the whole encrypted memory of a computing system, but do not have access to the encryption keys required for obtaining the corresponding plaintexts.

More formally, the input perturbing adversary is defined as follows: Let's consider a pair of encryption and decryption oracles E and D such that $D = E^{-1}$ and for which $D, E \in \mathbf{RO}^2(f, B, \epsilon)$ for some f, B, ϵ. An input perturbing adversary $M^D(q_0, \ldots, q_{m-1}, B)$ is defined as a polynomial time algorithm which:

- has oracle access to D
- has knowledge of m queries $q_0, \ldots, q_{m-1}$ to D and their responses $D(q_0), \ldots, D(q_{m-1})$, where the responses exhibit patterns: $D(q_0), \ldots, D(q_{m-1}) \in \mathbf{\Pi}(f)$
- can perform at most B non-repeating queries to D as part of the game, which are other than $q_0, \ldots, q_{m-1}$.

The algorithm succeeds if it finds a new input data word y which is different from $q_0, \ldots, q_{m-1}$, the output of which exhibits patterns, i.e., $D(y) \in \mathbf{\Pi}(f)$. The number of query-response values known m is assumed not to be large enough so as to leak information about the internals of E, D allowing, for instance, rainbow table attacks. The advantage of the input perturbing adversary is defined for the decryption operation as:

$$\begin{aligned} &\mathbf{Adv}(M^D(q_0, \ldots, q_{m-1}, B), f) = \\ &\text{Prob}[y \leftarrow M^D(q_0, \ldots, q_{m-1}, B); y \notin \{q_0, \ldots, q_{m-1}\}; \\ &D(y) \in \mathbf{\Pi}(f)] \end{aligned} \qquad (2)$$

B. Oracle replacing adversary

Another type of adversary, which we introduce in this paper is the 'oracle replacing adversary'. This adversary is associated with replay attacks. Replay attacks may happen across key domains such as network sessions that are encrypted with different keys, or encrypted memory domains. The model of the oracle replacing adversary can indeed be associated with such replay attacks under the assumption that the ingredient random permutations of an RO^2 construction use key values which are specific to particular domains of trust. When some valid encrypted data from one domain is replayed in another domain, then this replay attack is equivalent to replacing the ingredient random permutations of a RO^2 construction with another set of permutations, associated with a new domain, and observing the output.

More formally, lets consider a pair of encryption and decryption oracles E and D, $D = E^{-1}$ for which $D, E \in \mathbf{RO}^2(f, B, \epsilon)$ for some f, B, ϵ and have access to ingredient random permutations $R_0, R_1, R_2, \ldots$. An oracle replacing adversary $M^D(q_0, \ldots, q_{m-1}, B, \boldsymbol{R})$ is defined as a polynomial time algorithm which:

- has oracle access to $D^{R_0, R_1, R_2, \ldots}$
- has knowledge of m queries $q_0, \ldots, q_{m-1}$ to $D^{R_0, R_1, R_2, \ldots}$ and their responses $D(q_0), \ldots, D(q_{m-1})$, where the responses exhibit patterns: $D(q_0), \ldots, D(q_{m-1}) \in \mathbf{\Pi}(f)$
- has access to a set $\boldsymbol{R}$, $\boldsymbol{R} = \{\{R_0^{(0)}, R_1^{(0)}, \ldots\}, \ldots, \{R_0^{(n-1)}, R_1^{(n-1)}, \ldots\}\}$ of n sets of random permutations that have the same input and output length characteristics as $R_0, R_1, R_2, \ldots$ and are all different from $R_0, R_1, R_2, \ldots$
- can perform at most B non-repeating queries to $D^{R_0, R_1, R_2, \ldots}$ as part of the game, other than $q_0, \ldots, q_{m-1}$.

The algorithm succeeds if it finds a set of new ingredient random permutations $\{R'_0, R'_1, R'_2, \ldots\} \in \boldsymbol{R}$ and an input query word $y \in \{q_0, \ldots, q_{m-1}\}$, the output of which exhibits patterns when y is applied on an instance of D that uses the new ingredient random permutations $R'_0, R'_1, R'_2, \ldots$, i.e., $D^{R'_0, R'_1, R'_2, \ldots}(y) \in \mathbf{\Pi}(f)$. As in the case of the input perturbing adversary, this adversary also does not have any knowledge about possible keys used by E, D. The advantage of the oracle replacing adversary is defined for the decryption operation as:

$$\begin{aligned}
&\mathbf{Adv}(M^D(q_0, \ldots, q_{m-1}, B, \boldsymbol{R}), f) = \\
&\text{Prob}[\{\{R'_0, R'_1, \ldots\}, y\} \leftarrow M^D(q_0, \ldots, q_{m-1}, B, \boldsymbol{R}); \\
&\ y \in \{q_0, \ldots, q_{m-1}\};\ \{R'_0, R'_1, \ldots\} \in \boldsymbol{R}; \\
&D^{R'_0, R'_1, \cdots}(y) \in \mathbf{\Pi}(f)]
\end{aligned} \tag{3}$$

V. Main results

Our main results connect the concept of a random oracle according to an observer function with security claims associated with the input perturbing and oracle replacing adversary models. For the sake of completeness, we begin by stating the main result of reference [3], which concerns the security of an RO^2 construction in the input perturbing adversary model.

Theorem 1: *About the security of an* RO^2 *construction in the input perturbing adversary model.* Given a pair of encryption and decryption oracles E and D, $D = E^{-1}$, an observer function f, a query bound B, and an observation indistinguishability parameter ϵ such that: $D, E \in \mathbf{RO}^2(f, B, \epsilon)$ then for any input perturbing adversary $M^D(q_0, \ldots, q_{m-1}, B)$:

$$\mathbf{Adv}(M^D(q_0, \ldots, q_{m-1}, B), f) \leq P_f \cdot 2^\epsilon \tag{4}$$

We note that P_f is the pattern observation probability associated with observer function f. The proof of Theorem 1 can be found in reference [3].

A next theorem concerns the security of RO^2 constructions in the oracle replacing adversary model. It is in the proof of this theorem that we make use of the constraints of RO^2 constructions which we introduce in Figure 3 above.

Theorem 2: *About the security of an* RO^2 *construction in the oracle replacing adversary model.* Given a pair of encryption and decryption oracles E and D, $D = E^{-1}$, an observer function f, a query bound B, and an observation indistinguishability parameter ϵ such that: $D, E \in \mathbf{RO}^2(f, B, \epsilon)$ then for any oracle replacing adversary$M^D(q_0, \ldots, q_{m-1}, B, \boldsymbol{R})$:

$$\mathbf{Adv}(M^D(q_0, \ldots, q_{m-1}, B, \boldsymbol{R}), f) \leq P_f \cdot 2^\epsilon \tag{5}$$

where P_f is the pattern observation probability associated with observer function f.

Proof of Theorem 2: We need to show that for every oracle replacing adversary:

$$\begin{aligned}
&\text{Prob}[\{\{R'_0, R'_1, \ldots\}, y\} \leftarrow M^D(q_0, \ldots, q_{m-1}, B, \boldsymbol{R}); \\
&\ y \in \{q_0, \ldots, q_{m-1}\};\ \{R'_0, R'_1, \ldots\} \in \boldsymbol{R}; \\
&D^{R'_0, R'_1, \cdots}(y) \in \mathbf{\Pi}(f)] \leq P_f \cdot 2^\epsilon
\end{aligned} \tag{6}$$

We assume that an adversary exists for which the relation 6 does not hold. This adversary repeatedly succeeds in producing random permutation replacements and inputs y the outputs of which exhibit patterns with probability greater than $P_f \cdot 2^\epsilon$. We show that if such adversary exists then it is not possible for E, D to be $\mathbf{RO}^2(f, B, \epsilon)$ which contradicts our assumption. To prove Theorem 2, we first state and prove a lemma that bounds the probability of seeing patterns in the output of an RO^2 construction once we replace the ingredient random permutations.

Lemma 1: Let's assume that we have a pair of encryption and decryption oracles $D, E \in \mathbf{RO}^2(f, B, \epsilon)$ for some f, B, ϵ, $D = E^{-1}$, and some input y, such that $D^{R_0, R_1, R_2, \cdots}(y) \in \mathbf{\Pi}(f)$. Then for any set of ingredient random permutation replacements $R'_0, R'_1, \ldots$ which are also random permutations, the probability $\text{Prob}[D^{R'_0, R'_1, \cdots}(y) \in \mathbf{\Pi}(f)]$ of seeing patterns in the output of y is bounded by:

$$\begin{aligned}
&\text{Prob}[D^{R'_0, R'_1, \cdots}(y) \in \mathbf{\Pi}(f) \mid \\
&D^{R_0, R_1, R_2, \cdots}(y) \in \mathbf{\Pi}(f)] \leq P_f \cdot 2^\epsilon
\end{aligned} \tag{7}$$

Proof of Lemma 1: We consider a system 1 shown in Figure 4, where the decryption oracle D accesses the original

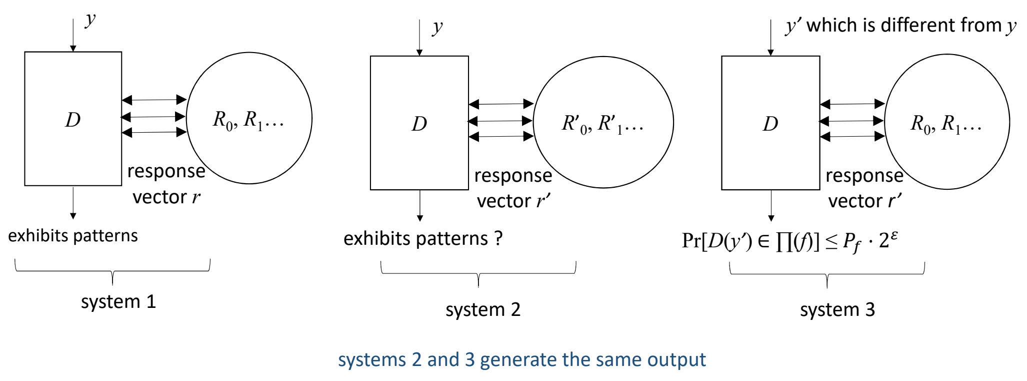

Fig. 4. Replacing ingredient random permutations inside an RO^2 construction

ingredient random permutations $R_0, R_1, R_2, \dots$ and in this system one particular query to $R_0, R_1, R_2, \dots$ returns a query response vector r. In another system, system 2, the original ingredient random permutations $R_0, R_1, R_2, \dots$ are replaced by $R'_0, R'_1, \dots$, and the same query now returns a different response vector r'. In system 3 of the figure, the decryption oracle D accesses the original ingredient random permutations $R_0, R_1, R_2, \dots$, but in this system the corresponding query to $R_0, R_1, R_2, \dots$ returns a response vector r', which is the same as the one returned by the permutations of system 2. As the ingredient random permutations $R_0, R_1, R_2, \dots$ are bijective functions, they are invertible. By inverting the ingredient random permutations $R_0, R_1, R_2, \dots$ on the response vector r', one computes an input y' which needs to be provided to system 3 in order for the ingredient random permutations of this system to return the same response vector r', which is returned in system 2. Due to $R_0, R_1, R_2, \dots$ being bijective and the RO^2 construction constraints introduced in Figure 3, input y' must be different from y. Since $y' \neq y$, and system 3 is an RO^2 construction, then the output of system 3 exhibits patterns with probability:

$$\text{Prob}[D^{R_0,R_1,\cdots}(y') \in \mathbf{\Pi}(f) \mid D^{R_0,R_1,R_2,\cdots}(y) \in \mathbf{\Pi}(f)] \le P_f \cdot 2^{\epsilon} \tag{8}$$

The proof of Lemma 1 completes by observing that systems 2 and 3 generate the same output. Hence:

$$\text{Prob}[D^{R'_0,R'_1,\cdots}(y) \in \mathbf{\Pi}(f) \mid D^{R_0,R_1,R_2,\cdots}(y) \in \mathbf{\Pi}(f)] = \text{Prob}[D^{R_0,R_1,\cdots}(y') \in \mathbf{\Pi}(f) \mid D^{R_0,R_1,R_2,\cdots}(y) \in \mathbf{\Pi}(f)] \le P_f \cdot 2^{\epsilon} \tag{9}$$

and Lemma 1 is proven. We proceed with the proof of Theorem 2 by stating and proving one more Lemma:

Lemma 2: Let's consider an ensemble of ingredient random permutation sets $\{R_0^{(i)}, R_1^{(i)}, \dots\}$, $i \ge 0$. Let's also consider constructions $D, E \in \mathbf{RO}^2(f, B, \epsilon)$ for some f, B, ϵ and $D = E^{-1}$. We further consider a set of input indices, $J_0 = 0$, $J_1 > J_0$, $J_2 > J_1, \dots$ for which $J_{i+1} - J_i \le B$ for all $i \ge 0$. Using the constructions E, D and the indices $J_0, J_1, \dots$, we define permutation swapping constructions E', D' as constructions that accept as input discrete sequences of values $y_0, y_1, \dots$, have infinite lifetime as opposed to bounded by B, and provide output which is obtained as follows:

$$E'(y_j) = E^{R_0^{(i)}, R_1^{(i)}, \cdots}(y_j) \quad \text{and} \quad D'(y_j) = D^{R_0^{(i)}, R_1^{(i)}, \cdots}(y_j)$$
$$\text{for all } y_j \text{ such that } J_i \le j < J_{i+1} \tag{10}$$

If E', D' are defined by equation 10, then for any input sequence $\tilde{y}_0, \tilde{y}_1, \dots$ to D' which is not repeating inside the index bounds defined by $J_0, J_1, \dots$ the following inequality is true:

$$\text{Prob}[D'(\tilde{y}) \in \mathbf{\Pi}(f)] \le P_f \cdot 2^{\epsilon} \tag{11}$$

The fact that input sequence $\tilde{y}_0, \tilde{y}_1, \dots$ to D' is not repeating inside the index bounds means that $\tilde{y}_{j'} \neq \tilde{y}_{j''}$ for all $J_i \le j' < J_{i+1}$, $J_i \le j'' < J_{i+1}$, $j' \neq j''$ and $i \ge 0$. Relation 11 holds for every input $\tilde{y} \in \{\tilde{y}_0, \tilde{y}_1, \dots\}$.

Proof of Lemma 2: From the definition of equation 10 it is evident that E' and D' are also encryption and decryption oracles and that $D' = E'^{-1}$. The inputs to decryption oracle D' can either result in outputs with patterns or not. On the other hand, the inputs that result in patterns can be split into repeating inputs and non-repeating inputs, as inputs from the sequence $\tilde{y}_0, \tilde{y}_1, \dots$ to D' may be repeating across index bounds. It is sufficient to show that the property $\text{Prob}[D'(\tilde{y}) \in \mathbf{\Pi}(f)] \le P_f \cdot 2^{\epsilon}$ holds for the repeating inputs which produce at least one output with patterns. This is because the probability $\text{Prob}[D'(\tilde{y}) \in \mathbf{\Pi}(f)]$ is trivially 0 if it is known that inputs are always unsuccessful. On the other hand, for the non-repeating inputs the property does hold, as the constructions defined by E', D' are RO^2 inside the index bounds.

It is easy to see that, for the case of repeating inputs that produce at least one output with patterns, the property $\text{Prob}[D'(\tilde{y}) \in \mathbf{\Pi}(f)] \le P_f \cdot 2^{\epsilon}$ holds due to Lemma 1. Indeed let's consider some input $\tilde{y}_j$ such that $J_i \leq j < J_{i+1}$ for some $i \geq 0$. Let's also consider that this input, if passed to the decryption oracle D', produces output that exhibits patterns:

$$D'(\tilde{y}_j) = D^{R_0^{(i)}, R_1^{(i)}, \cdots}(\tilde{y}_j) \in \mathbf{\Pi}(f) \tag{12}$$

If this input $\tilde{y}_j$ appears in the input sequence again, outside the index bounds J_i and J_{i+1}, then the probability of seeing patterns at the output of this repeated instance of $\tilde{y}_j$ is bounded according to Lemma 1. Specifically, if value $\tilde{y}_j$ appears again inside the index bounds $J_{i'}$ and $J_{i'+1}$ for some $i' \neq i$, then the output of the decryption oracle D' for this repeated instance is equal to $D^{R_0^{(i')}, R_1^{(i')}, \cdots}(\tilde{y}_j)$. If we apply Lemma 1 to the sets of ingredient random permutations $\{R_0^{(i)}, R_1^{(i)}, \ldots\}$ and $\{R_0^{(i')}, R_1^{(i')}, \ldots\}$ and to the input value $\tilde{y}_j$, we obtain inequality:

$$\begin{aligned} &\text{Prob}[D^{R_0^{(i')}, R_1^{(i')}, \cdots}(\tilde{y}_j) \in \mathbf{\Pi}(f) \mid \\ &D^{R_0^{(i)}, R_1^{(i)}, \cdots}(\tilde{y}_j) \in \mathbf{\Pi}(f)] \leq P_f \cdot 2^{\epsilon} \end{aligned} \tag{13}$$

which completes the proof of Lemma 2.

We note that the inequality 11 of Lemma 2 holds even if the event $D'(\tilde{y}) \in \mathbf{\Pi}(f)$ is conditioned upon inputs to D' which are different from $\tilde{y}$. This is due to two facts. First, that the construction D' is RO^2 inside the index bounds. Second, that the probability of seeing patterns in the output of one life time of D (i.e., one set of index bounds of D') is conditionally independent of inputs that appear in other life times of D, where different sets of ingredient random permutations may be queried. The proof of this property is similar to the proof of Lemma 1 and omitted for conciseness.

Completing the proof of Theorem 2: Any instance of D that queries ingredient random permutations from one of the sets of $\boldsymbol{R}$ is RO^2 by the definition of D and due to the fact that the permutations contained in the sets of $\boldsymbol{R}$ are random permutations. Similarly, any permutation swapping construction D' which is produced from D and has infinite lifetime, exhibits patterns in its output with probability which is bounded according to Lemma 2, provided that the input is non-repeating inside index bounds.

Now, let's suppose we have an oracle replacing adversary M, which is repeatedly successful with probability higher than the bound $P_f \cdot 2^{\epsilon}$ of relation 6. In every attack, this adversary succeeds in computing a different y value and a different set of ingredient random permutations from $\boldsymbol{R}$ all resulting in patterns with probability $> P_f \cdot 2^{\epsilon}$. Furthermore, as in the proof of Theorem 1 [3], this adversary M can be turned into another adversary M' which always returns some output and is still successful in attacking D with probability $> P_f \cdot 2^{\epsilon}$. We consider that such attacks are repeated again and again. One can see that the attacks performed by adversary M' form a trace of queries to a permutation swapping construction D' as defined in Lemma 2. The expected value E of the ratio of successful queries to D' over all queries made needs to satisfy the inequality $E > P_f \cdot 2^{\epsilon}$, due to the assumption about the existence of the adversary M'. On the other hand, the same expected value needs to satisfy the inequality $E \leq P_f \cdot 2^{\epsilon}$ due to the fact that Lemma 2 holds, which is not possible. Hence, Theorem 2 is proven.

VI. Discussion

Our results reduce the proofs which establish security in the input perturbing and oracle replacing adversary models to showing that a cryptographic construction is in the class RO^2. The implication from Theorems 1 and 2 is that no matter how the adversary corrupts the ciphertext the probability that the patterns are visible in some plaintext can be bounded if a construction is in this class.

Our analysis encourages further study on cryptographic constructions that are RO^2. Some of these constructions are quite practical and inexpensive to build. For example, reference [3] introduces a construction called IVP, which is a three level confusion diffusion network that supports 32-bit implicit integrity associated with the 4 among 32 16-bit word equality pattern, the life time value of $B = 2^{32}$ queries and the indistinguishability parameter $\epsilon = 2.651$ bits. This construction successfully defends against on-line corruption and replay attacks on content which is 512 bits long. The overhead of the construction over a standard four block mode of AES (e.g., AES-XTS) is only two additional AES rounds in the data path, which can be minimal.

VII. Acknowledgement

The concept of implicit integrity has been proposed by David Durham and Men Long at Intel Labs [1]. The author would like to thank all members of the implicit integrity team at Intel Labs for their contributions to this effort.

References

[1] D. Durham and M. Long, *Memory Integrity*, United States Patent, No. 9,213,653, Decednber 2013.

[2] M. Kounavis, D. Durham, S. Deutsch, and S. Komijani, *Implicit Data Integrity: Protecting User Data without MACs*, SECRYPT 2018.

[3] M. Kounavis, D. Durham, S. Deutsch, S. Komijani, A. Papadimitriou, and K. Grewal, *There is No Need to Waste Communication Bandwidth on MACs*, IEEE GIIS 2018.

[4] *Secure Hash Standard*, Federal Information Processing Standards Publication FIPS PUB 180-4.

[5] *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*, Federal Information Processing Standards Publication FIPS PUB 202.

[6] *The Keyed-Hash Message Authentication Code (HMAC)*, Federal Information Processing Standards Publication FIPS PUB 198-1.

[7] *SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash and ParallelHash*, NIST Special Publication 800-185.

[8] V. T. Hoang, T. Krovetz and P. Rogaway, *Robust Authenticated Encryption: AEZ and the Problem that it Solves*, EUROCRYPT 2015.

[9] C. Badertscher, C. Matt, U. Maurer, P. Rogaway and B. Tackmann, *Robust Authenticated Encryption and the Limits of Symmetric Cryptography*, 15th IMA International Conference on Cryptography and Coding, 2015.

[10] U. Maurer, R. Renner and C. Holenstein, *Indifferentability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology*, In Moni Naor, editor, First Theory of Cryptography Conference - TCC 2004, volume 2951 of LNCS, pages 21-39. Springer-Verlag, February 1921 2004.